CORDIC Implementation for Ultralow Power Applications

Patricia L. Gonzalez

***Abstract*—We present a 15.9pJ implementation of the generalized COordinate Rotation DIgital Computer (CORDIC) algorithm for an ultralow power and flexible Body Sensor Node (BSN). This accelerator allow us to relax the voltage-frequency operation point to 0.5V-300KHz without affecting the latency of the node, by calculating basic mathematical functions in only 15 clock cycles.**

***Index Terms*—Biomedical Signal Processing, Subthreshold.**

I. Introduction

BODY Sensor Nodes (BSNs) are wearable devices that promise to enhance the assessment of human physiology and performance [1]. These systems are able to sense, process, store and transmit reliable physiological data. A BSN should be small enough that the user can barely notice it. This constraint severely limits the processing capabilities and more importantly the power budget. Typically around 80% of the energy consumed by a BSN is used for streaming raw data to an aggregator [2]. However, under energy constrained scenarios Zhang et. al. [3] showed that the use of on node digital signal processing and selective transmission of only relevant information can reduce the power consumption due to wireless transmission to 0.013%. With this approach, energy consumption optimization, shifts to the other components of the system as the digital processing data paths. Ickes et. al [4], showed that one way of saving energy consumption is using Digital Signal Processors (DSP) as accelerators instead of Microcontrollers. The author demonstrated savings in energy consumption of up to 11 times compared to software based implementations.

These energy savings, motivates us to develop a hardware implementation of the generalized CORDIC algorithm, as part of our highly flexible, bateryless, therefore ultralow power, SoC platform. This platform supports multiple sensing modalities, extracts information from data flexibly across applications, harvests and delivers power efficiently, and communicates wirelessly [5]. This CORDIC accelerator, is able to calculate $\sin(x)$, $\cos(x)$, $\arctan(y/x)$, $\sinh(x)$, $\cosh(x)$, $\tanh(yx)$, e^x, $ln(x)$, $1x$, $\sqrt{x^2+y^2}$ and $\sqrt{x}$. All this basic mathematical functions enable the node for many different applications. For example, to process Electrocardiogram (ECG) features, the algorithm requires the computation of $ln(x)$ [6]; while the gait speed estimation requires $\sqrt{x^2+y^2}$ [7].

In this paper we present a hardware implementation of the generalized CORDIC algorithm. This accelerator, is able to calculate up to 11 basic mathematical functions within the limited energy budget of a bateryless BSN. This paper, starts with a description of the basic CORDIC algorithm in Section II. Section III describes our hardware implementation. Finally, Section IV discusses the results.

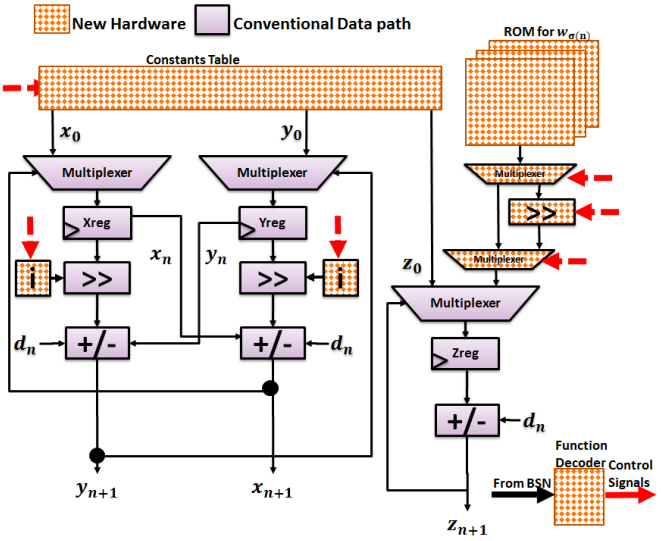

Fig. 1. CORDIC Modified to support different operation modes.

II. CORDIC Algorithm

The CORDIC algorithm was proposed in 1959 by Jack E. Volder [8]. It uses simple adders and shifters, which makes it an attractive alternative for hardware implementations when basic mathematical functions are required. The algorithm was originally developed to calculate $\sin(\alpha)$ and $\cos(\alpha)$ as a sequence of positive and negative elementary angle rotations, and is described by the iteration in the first column of Tab. I. The terms $\tan^{-1}2^{-n}$ represents the rotation at each step

Iteration	Input	Output
$x_{n+1} = x_n - d_n y_n 2^{-n}$ $y_{n+1} = y_n + d_n x_n 2^{-n}$ $z_{n+1} = z_n - d_n \tan^{-1} 2^{-n}$ $d_n = \begin{cases} 1 & \text{if } z_n > 0 \\ -1 & \text{otherwise} \end{cases}$	$x_0 = 1 \div K$ $y_0 = 0$ $z_0 = \alpha$ $K = \prod_i \sqrt{1+2^{-2i}}$	$x_n = \cos\alpha$ $y_n = \sin\alpha$ $z_n = 0$

TABLE I
Basic Volder's algorithm for circular iteration type.

and are precomputed and stored in memory. d_n is 1 or -1 depending of the sign of z_n and identifies the direction of the rotation at each step. If x_0, y_0 and z_0 are initialized as described in the second column of Tab. I, then x_n, y_n and z_n converge to $\sin\alpha$ and $\cos\alpha$ as described in the third column of Tab. I. This is known as the rotation mode of the algorithm

Type	m	w_n	Rotation Mode $d_n = sign(z_n)$	Vectoring Mode $d_n = -sign(y_n)$
Circular	1	$\tan^{-1} 2^{-k}$	$x_n \rightarrow K(x_0 \cos z_0 - y_0 \sin z_0)$ $y_n \rightarrow K(y_0 \cos z_0 - x_0 \sin z_0)$ $z_n \rightarrow 0$	$x_n \rightarrow K\sqrt{x_0^2 + y_0^2}$ $y_n \rightarrow 0$ $z_n \rightarrow z_0 + \arctan y_0/x_0$
Linear	0	2^{-k}	$x_n \rightarrow x_0$ $y_n \rightarrow y_0 + x_0 z_0$ $z_n \rightarrow 0$	$x_n \rightarrow x_0$ $y_n \rightarrow 0$ $z_n \rightarrow z_0 + y_0/x_0$
Hyperbolic	-1	$\tanh^{-1} 2^{-k}$	$x_n \rightarrow K'(x_0 \cosh z_0 - y_0 \sinh z_0)$ $y_n \rightarrow K'(y_0 \cosh z_0 - x_0 \sinh z_0)$ $z_n \rightarrow 0$	$x_n \rightarrow K'\sqrt{x_0^2 - y_0^2}$ $y_n \rightarrow 0$ $z_n \rightarrow z_0 + \tanh^{-2} y_0/x_0$

TABLE II
SUMMARY OF THE GENERALIZED CORDIC ALGORITHM.

since it is rotating an angle $z_0 = \alpha$ by $d_n \arctan 2^{-n}$. Similarly, CORDIC can also start with a vector which is rotated until it becomes parallel to the x axis, which is called the vectoring mode. Furthermore, a slight modification to the basic algorithm can be included to extend it to the hyperbolic and linear coordinate systems as well [9]. The generalized CORDIC algorithm that includes the rotation and vectorial mode, as well as the circular, hyperbolic and linear coordinate systems is described in equations 1- 3. Tab. II shows the values of d_n, m, and w_n and the different functions that can be calculated with each operation mode.

$$x_{n+1} = K_{mi}(x_n - m d_n y_n 2^{-(n)}) \quad (1)$$

$$y_{n+1} = K_{mi}(y_n + d_n x_n 2^{-(n)}) \quad (2)$$

$$z_{n+1} = z_n - d_n w_{(n)} \quad (3)$$

III. HARDWARE IMPLEMENTATION

Typical hardware implementations focus only in one mode of operation, since they target very specific applications. Fig. 1 shows the block diagram of the CORDIC accelerator. The typical CORDIC data-path is composed by three accumulator registers x_n, y_n, z_n, two adders and three shifters, represented by the not-filled blocks. In this implementation, we add the hardware dotted to support the generalized CORDIC algorithm, reusing the same data path for the rotation and vectoring mode as well as for circular, linear and hyperbolic coordinate systems [10]. To decrease the power consumption during the idle time of the block we include power gate and clock gate. Power gating, uses a large sleep transistors to switch of the power supply rail and reduce the leakage of the circuit. Clock Gating, avoids unnecessary switching of the circuits to decrease dynamic consumption. These two features are controlled by a programmable digital power manager in the BSN that power/clock gates the accelerator based on the available energy and the application. The following subsections, explain the hardware modifications highlighted in Fig. 1.

A. Function Decoder

The function decoder, bottom-right corner in Fig. 1, receives the data from the node, and prepares the accelerator to start the iteration. It receives 16 bits as the input. The 4 most significant bits represent a $function_{id}$ and the 12 at the right are the input data. The decoder uses the $function_{id}$ to identify the appropriate values for d_n, m, w_n, x_0, y_0, and z_0 which vary with the rotation type and coordinate system. Finally, it selects one of the three ROMs that storage the pre-calculated w_n for each coordinate system.

Operation	Conventional CORDIC	Proposed Enhancement
$\sqrt{x}$	$0.0267 < x < 2.339$	$0.00097 < x < 20$
$\ln x$	$0.1068 < x < 9.35$	$0.00101 < x < 31$
$\exp x$	$-1.74 < x < 1.74$	$-3.44 < x < 3.44$

TABLE III
CONVERGENCE RANGE SUPPORTED BY THIS IMPLEMENTATION.

B. Incrementing the range of convergence

The main concern of a CORDIC implementation, is the limited range of convergence of the algorithm. Following the approach described in [11], we include non-positively indexed iterations to the basic algorithm. Also, in the case of the hyperbolic system we slightly modify w_n. so for $i < 0$ it is $\tanh^{-1}(1 - 2^{k-2})$ instead of $\tanh^{-1} 2^{-k}$. Tab. III shows the original and incremented range of convergence.

C. Error Handling

The CORDIC algorithm is affected by three primary sources of error:

1) Rounding error: The rounding error is due to the truncation of CORDIC internal variables by the finite length of storage elements. To mitigate this effect we added 4 guard bits to the 12 bits at the input, obtaining a 16 bits width data path. Besides the guard bits, we also included a mechanism to adjust the binary point as explained later in this section.

2) Angle Approximation Error: In theory the rotation angle is decomposed into infinite number of elementary angles, however, for a practical implementation only a finite number of micro rotations is considered causing the angle approximation error. For an energy perspective, rather a small number of micro rotations is desirable, for this design, we kept only as many iterations as the ones supported by our 16 bits data-path width. The maximum error obtained was less than 10^{-2}.

3) Range of Convergence Increment: When increasing the range of convergence we are also incrementing error in the calculation. To address this we included the capability of adjusting the binary point and restricting the increment to the number of iterations only when the range of data requires it. For that we use a reserved $function_{id}$ that indicates to the function decoder which fixed point data representation ($Q_{n,m}$: n integers, m fractional) to be used from three different options, $Q_{6,10}$, $Q_{4,12}$ or $Q_{3,13}$. A shifter at the output of the tables and the initialization variables allow us to adjust the hardwired constants to the different representations.

Fig. 2 shows the error calculated for $ln(x)$ in all the range of convergence, for the different data formats available. For $Q_{6,10}$ we increased the range of convergence up to 31 but pay the penalty in an error increase. However if the range of the

	MCU [3]	CORDIC [3]	This work
Total energy	279nJ	1.3nJ	15.9pJ
Cycle Count	3495	50	13
Datapath Width	16	24 bits	16 bits
Operating Voltage	1V	1V	0.5V
Frequency of Operation	10MHz	10MHz	348KHz

TABLE IV
COMPARISON CHART FOR RELEVANT METRICS OF THE DESIGN

data is less, the block can be configured to operate in $Q_{4,12}$ or $Q_{3,13}$ reducing the number of extra iterations, incrementing the number of bits for precision therefore decreasing the error.

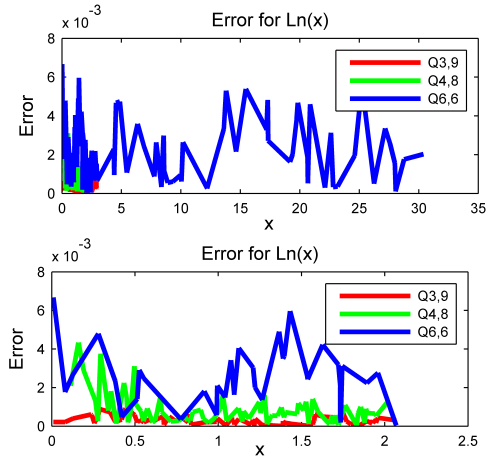

Fig. 2. Error for Ln(x). (Above) All convergence range. (Below) Magnification where convergence range overlaps for all the data formats.

IV. RESULTS

The CORDIC accelerator was synthesized and fabricated as part of a BSN node in a low leakage 130 μm CMOS process using a library optimized for sub-threshold operation. Fig. 3 shows the simulated Power, Energy and Delay vs VDD for this accelerator. The optimum operating point to minimize Energy per cycle is around 0.5V @ 348KHz with a power consumption of 381nW and Energy per cycle of 1pJ.

Although there is a lot of literature describing hardware implementations for CORDIC, very little comparable work has been published. This work consumes 3X less energy than the lowest published implementation, see Tab. IV, by keeping a data path limited to 16 bits and using the smallest number of iterations required per operation. This combination allows us to lower the VDD and frequency of operation to the minimum energy point and still meet the latency constraints imposed by the processing of bio signals.

V. CONCLUSION

This paper presents an ultralow power CORDIC accelerator able to calculate up to 11 different basic mathematical functions at an energy budget of 15pJ per operation. To implement this accelerator we use a sub-threshold optimized library, power gating and clock gating techniques, a small data path width plus a small number of iterations to be able to operate at

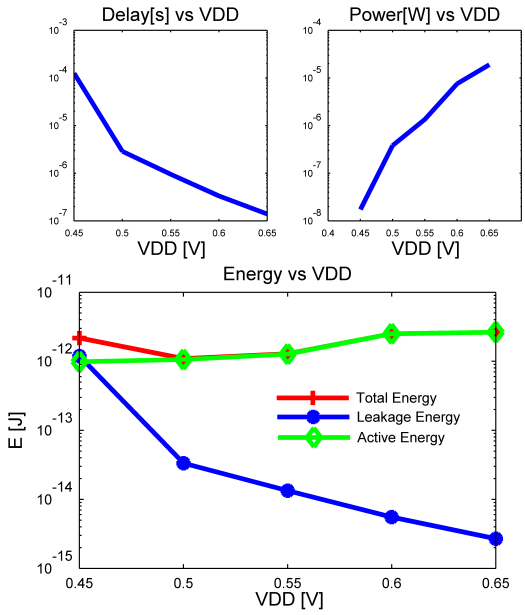

Fig. 3. Delay, Power and Energy vs VDD

the minimum energy point of the accelerator and still be able to meet latency constrains for bio signal processing. Although a small datapath and reduced number of iterations causes an increment in the error we believe it is still fully functional for the targeted applications.

REFERENCES

[1] B. Calhoun, J. Lach, J. Stankovic, D. Wentzloff, K. Whitehouse, A. Barth, J. Brown, Q. Li, S. Oh, N. Roberts, and Y. Zhang, "Body sensor networks: A holistic approach from silicon to users," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 91–106, Jan 2012.

[2] "Energy fidelity scalability," http://wirelesshealth.virginia.edu/node/35, accessed: 2015-01.

[3] Y. Zhang, F. Zhang, Y. Shakhsheer, J. Silver, A. Klinefelter, M. Nagaraju, J. Boley, J. Pandey, A. Shrivastava, E. Carlson, A. Wood, B. Calhoun, and B. Otis, "A batteryless 19 u w mics/ism-band energy harvesting body sensor node soc for exg applications," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 1, pp. 199–213, Jan 2013.

[4] N. Ickes, D. Finchelstein, and A. Chandrakasan, "A 10-pj/instruction, 4-mips micropower dsp for sensor applications," in *Solid-State Circuits Conference, 2008. A-SSCC '08. IEEE Asian*, Nov 2008, pp. 289–292.

[5] A. Klinefelter, N. E. Roberts, Y. Shakhsheer, P. Gonzalez, A. Shrivastiva, A. Roy, K. Craig, M. Faisal, J. Boley, S. Oh, Y. Zhang, D. Akella, D. D. Wentzloff, and B. H. Calhoun, "A 6.45 w self-powered iot soc with integrated energy-harvesting power management and ulp asymmetric radios," in *ISSCC*, San Francisco, CA, 02/2015 2015.

[6] W. Zong, G. Moody, and D. Jiang, "A robust open-source algorithm to detect onset and duration of qrs complexes," in *Computers in Cardiology, 2003*, Sept 2003, pp. 737–740.

[7] S. Chen, C. Cunningham, J. Lach, and B. Bennett, "Extracting spatio-temporal information from inertial body sensor networks for gait speed estimation," in *Body Sensor Networks (BSN), 2011 International Conference on*, May 2011, pp. 71–76.

[8] J. E. Volder, "The cordic trigonometric computing technique," *Electronic Computers, IRE Transactions on*, vol. EC-8, no. 3, pp. 330–334, Sept 1959.

[9] D. Henderson, "Elementary functions: Algorithms and implementation," *Mathematics and Computer Education*, vol. 34, no. 1, p. 94, 2000.

[10] J. Kwong and A. Chandrakasan, "An energy-efficient biomedical signal processing platform," in *ESSCIRC, 2010 Proceedings of the*, Sept 2010, pp. 526–529.

[11] X. Hu, R. Harber, and S. Bass, "Expanding the range of convergence of the cordic algorithm," *Computers, IEEE Transactions on*, vol. 40, no. 1, pp. 13–21, Jan 1991.